Proving Normal Form Formulas by Clause Expansion

Wenhui Zhang
Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
P.O.Box 8718, Beijing 100080, China
zwh@ios.ac.cn

December 8, 2003

Abstract

Two important methods for the verification of the validity or unsatisfiability of propositional formulas are the Davis-Putnam procedure (DPLL) and resolution. The advantage of DPLL is that it does not produce redundancy in the sense that all intermediate results are useful in forming a proof of validity or unsatisfiability. On the other hand, resolution may produce much shorter proofs when the parent clauses of the resolution rule are correctly selected in each step of resolution. The problem of the latter is the difficulty of choosing correct pairs of parent clauses and produce good resolvents. We propose an approach also based on producing new clauses (clause expansion) to supplement these approaches. A first order extension of the clause expansion proof systems is given at the end of this paper.

1 Introduction

Two important methods for the verification of the validity or unsatisfiability of propositional formulas are the Davis-Putnam procedure (DPLL) [9, 8] and resolution [17, 5]. The advantage of DPLL is that it does not produce redundancy in the sense that all intermediate results are useful in forming a proof of validity or unsatisfiability. On the other hand, resolution may produce much shorter proofs when the parent clauses of the resolution rule are correctly selected in each step of resolution. The problem of resolution is that there are normally many pairs of clauses that can be used as parent clauses and it is not easy to make the right choices of parent clauses and produce good resolvents (i.e. new clauses produced from pairs of parent clauses). The length of clauses is a strong criterion for usefulness of a resolvent, since the goal of resolution is the empty clause. However, in most resolution steps, it is not possible to produce a shorter clause from existing ones, and the length of clauses is therefore not very practical as an assessment criterion. In general, it is difficult to assess the usefulness of a resolvent and as a result, many redundant resolvent may be produced during the resolution process. Instead of looking at the usefulness of resolvents, there has been a lot of effort to limit the number of producible resolvents by requiring that potential parent clauses must satisfy given criteria. Some of the techniques have succeeded in limiting the number of producible resolvents at each step. However such techniques normally leads to the loss of capability of constructing short proofs. For instance, there are formulas that have linear size resolution proofs, but requiring exponential size proofs in negative resolution [3]. We propose an approach also based on producing new clauses (clause expansion) to supplement these approaches. The approach is capable of constructing shorter proofs compared to DPLL, and on the other hand, it could be

easier to make guidelines for assessing the usefulness of new clauses when compared to resolution. Comparing with resolution, the approach also has better potential for utilizing parallel computing power.

Preliminaries

The basic syntactic units of propositional logic are proposition symbols representing atomic propositions which may have values either $True$ or $False$. Proposition symbols are combined with logical connectives to form composite sentences, called formulas. The most frequently used logical connectives include $\neg, \wedge, \vee$ and $\supset$ where $\neg$ is the unary connective for negation, $\wedge, \vee$ and $\supset$ are binary connectives for respectively conjunction, disjunction and implication. Let the set of proposition symbols be $\mathcal{P} = \{A_1, A_2, ...\}$. The set of propositional formulas is inductively defined by stating that every proposition symbol is a formula (called atomic formulas), and that if φ and ψ are formulas, then $(\neg\varphi), (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \supset \psi)$ are formulas. Atomic formulas and the negation of such formulas are called literals. The negation of a literal is also called the complement of the literal.

We consider disjunctive normal form (DNF) formulas. A formula is in DNF, if it is a disjunction of conjunctions of literals. A formula can be transformed to a DNF-formula using the following equivalences:

$\varphi \supset \psi$	$\equiv$	$\neg(\varphi \wedge \neg\psi)$
$\neg(\varphi \wedge \psi)$	$\equiv$	$\neg\varphi \vee \neg\psi$
$\neg(\varphi \vee \psi)$	$\equiv$	$\neg\varphi \wedge \neg\psi$
$\neg\neg\varphi$	$\equiv$	φ
$\varphi \wedge (\psi_0 \vee \psi_1)$	$\equiv$	$(\varphi \wedge \psi_0) \vee (\varphi \wedge \psi_1)$
$(\varphi_0 \vee \varphi_1) \wedge \psi$	$\equiv$	$(\varphi_0 \wedge \psi) \vee (\varphi_1 \wedge \psi)$

We call the conjunction of literals a *clause*. The order of literals in a clause does not matter. A DNF-formula can be represented as a set of clauses. In the following, we also call a set of clauses a DNF-formula.

2 The Proof System

Similar to resolution, our proof system for DNF-formulas is based on producing clauses, however, the new clauses of this proof system is produced by expanding what we already have proved, not from the set of input clauses. The rules are as follows:

Axiom: $$A, \neg A$$

Expansion: $$\frac{\Delta, \varphi}{\Delta, A \wedge \varphi, \neg A \wedge \varphi}$$

Simplification: $$\frac{\Delta, \varphi \wedge \psi}{\Delta, \varphi}$$

Weakening: $$\frac{\Delta}{\Delta, \varphi}$$

We call this system for CE (clause expansion). Note that we do not distinguish between $\varphi \wedge \psi$ and $\psi \wedge \varphi$, and therefore there is no need for a left and a right simplification rule.

A proof system is sound if every rule of the system maps a (possibly empty) set of valid formulas to a valid formula (in our case, the representation of a formula is a set of clauses). The rules of CE are obviously sound.

A proof system is complete if every valid formula is provable in the system. The completeness of CE for DNF-formulas can be established by the following arguments: Suppose that we have a valid DNF-formula $\{c_1, ..., c_n\}$ with $A_1, ..., A_n$ as the only proposition symbols. By using the axiom and the expansion rule, we can produce the set of the $N = 2^n$ distinct clauses $\{v_1 \wedge \cdots \wedge v_n \mid v_i \in \{A_i, \neg A_i\}\}$. Let the clauses be denoted by $d_1, ..., d_N$. Since $\{c_1, ..., c_k\}$ is valid, each d_i must be an extension of some c_j. By the simplification rule, we obtain either $\{c_1, ..., c_k\}$ or a proper subset of $\{c_1, ..., c_k\}$. In the latter case, we extend the subset to $\{c_1, ..., c_k\}$ using the weakening rule.

Theorem 2.1 CE *is sound and complete for DNF-formulas.*

2.1 Goal Oriented Proofs

The proof rules work towards different directions. For instance, the axiom may be instantiated to be $A, \neg A$ with any atomic formula A. Given a DNF-formula to prove, we should base our choice of the instances of the axiom and other rules on criteria related to the DNF-formula. To make the reasoning of the choice explicit, we add the goal into the proof rules and modify our system to be as follows:

Axiom: $$\Lambda \quad ; \quad A, \neg A$$

Expansion: $$\frac{\Lambda \quad ; \quad \Delta, \varphi}{\Lambda \quad ; \quad \Delta, A \wedge \varphi, \neg A \wedge \varphi}$$

Simplification: $$\frac{\Lambda \quad ; \quad \Delta, \varphi \wedge \psi}{\Lambda \quad ; \quad \Delta, \varphi}$$

Absorption: $$\frac{\Lambda, \varphi \quad ; \quad \Delta, \varphi}{\Lambda, \varphi \quad ; \quad \Delta}$$

Let us denote this system by $\mathrm{CE_G}$ (goal-oriented CE). Λ is the intended proof goal. A proof of Λ is completed, when we reach a conclusion of the form "Λ;", i.e. when the set on the right hand side of the semicolon is empty. The axiom of $\mathrm{CE_G}$ can be considered as a combination of the axiom of CE and the weakening rule. Therefore there is no need for a weakening rule in $\mathrm{CE_G}$. The absorption rule can be considered as a contraction that combines two copies of a formula into one. The expansion rule and the simplification rule are similar to those of CE.

The simplification rule and the absorption rule can be combined to formed a derived absorption rule as follows:

$$\frac{\Lambda, \varphi \quad ; \quad \Delta, \varphi \wedge \psi}{\Lambda, \varphi \quad ; \quad \Delta}$$

Note that ψ in the derived absorption rule could be the empty clause and the absorption rule is a special case of the derived absorption rule. In practice, we shall use the derive absorption rule whenever it is applicable, however, we keep the formal system (not replacing the absorption rule with the new one) for simplicity. Let $L(\Lambda)$ be set of literals that appear in Λ. Some guidelines for the rules are as follows:

- The axiom is used only when both A and $\neg A$ are in $L(\Lambda)$.
- The absorption rule is given the highest priority and is applied whenever it is applicable.
- The simplification rule is given priority, if some clause can be simplified to an existing clause (either on the left or right hand side of the semicolon).
- The expansion rule is used only when both A and $\neg A$ are in $L(\Lambda)$.

By studying the relation between the clauses, we can formulate situation specific guidelines. For instance, it is a good strategy to expand a formula $\varphi \wedge \psi$ on a literal L, if L is not used before and $\varphi \wedge L$ or $\varphi \wedge \neg L$ is in Λ. This strategy followed by an application of the absorption rule could be formulated as a derived rule:

$$\frac{\Lambda, \varphi \wedge L \quad ; \quad \Delta, \varphi \wedge \psi}{\Lambda, \varphi \wedge L \quad ; \quad \Delta, \varphi \wedge \psi \wedge \neg L}$$

There has been a lot of research on strategies for choosing a literal to be branched on for DPLL or to be resolved upon for resolution (see e.g. [14, 21, 10]). Deciding whether a literal is optimal for branching is NP-hard [15]. The above rule represents a strategy for choosing a literal to expand in the given special case.

3 Application Examples

We presents examples to how the proof system $\mathrm{CE_G}$ works. The first one is a graph-based example [18, 19]. It shows that DPLL does not p-simulate (the reader is referred to [6] for a discussion of the concepts p-simulation and p-equivalence). The second one is a proof of the mutilated chessboard problem [7]. It provides a matching proof length to the lower-bound of the resolution refutations of the problem [7]. The third one is a proof of the pigeonhole principle [11]. It provides a slightly lower upper bound than an upper bound for proving the pigeonhole principle by resolution reported in [4] and shows that the proof system may be simpler to use than resolution in some cases. The fourth example is based on implication graphs which were used to prove that linear resolution and negative resolution do not p-simulate unrestricted resolution [3]. We provide a proof of formulas based on such graphs by $\mathrm{CE_G}$ in the number of steps linear to the size of the formulas.

3.1 Charged Graphs

Let G be a graph. Let a labeled graph G' be G with the edges labeled with distinct literals that are not complement of each other. Let each vertex x be associated with an assignment $\mathrm{Charge}(x) \in \{0, 1\}$. If $l_1, ..., l_n$ are the literals labeling the edges attached to x, then $\mathrm{Clauses}(x)$ is the set of clauses of the form $l'_1 \vee \cdots \vee l'_n$ such that $l'_i \in \{l_i, \neg l'_i\}$ and the parity of the number of complemented literals of $l_1, ..., l_n$ in each of the clauses is opposite to $\mathrm{Charge}(x)$. The set of clauses Clauses(G') is the union of all the sets $\mathrm{Clauses}(x)$ for x a vertex in G. $\mathrm{Charge}(G')$ is the sum of charges on the vertices of G modulo 2.

Let G_n be a labeled graph consists of 2^n vertices $v_1, ..., v_{2^n}$ with $\mathrm{Charge}(G_n)$=1 and with adjacent vertices v_i and v_{i+1} joined by n edges. The set of clauses $\mathrm{Clauses}(G_n)$ contains $2^{3n}(\frac{1}{2} - O(2^{-n}))$ clauses of size at most $2n$. The negation of $\mathrm{Clauses}(G_n)$ can be proved by $\mathrm{CE_G}$ with $O(2^{3n})$ steps as follows:

- There are n literals attached to v_1. Let the literals be $L_{11}, ..., L_{1n}$. By a full expansion on these n literals, we obtain 2^n clauses, of which 2^{n-1} can be absorbed into the negation of Clauses(G_n).

- There remain 2^{n-1} clauses. There are additional n literals attached to v_2. Let the literals be $L_{21}, ..., L_{2n}$. By expanding these n literals for each of the remaining clauses, we obtain $2^{n-1} \cdot 2^n$ clauses, of which $2^{n-1} \cdot 2^{n-1}$ can be absorbed into the negation of Clauses(G_n).

- There remain $2^{n-1} \cdot 2^{n-1}$ clauses. By using the simplification rule to eliminate $L_{11}, ..., L_{1n}$ and their complementary literals from the remaining clauses, we obtain 2^{n-1} clauses over literals $L_{21}, ..., L_{2n}$ and their complements.

- Repeating step 2 and step 3 by replacing
  $L_{21}, ..., L_{2n}$ with $L_{31}, ..., L_{3n}$ and
  $L_{11}, ..., L_{1n}$ with $L_{21}, ..., L_{2n}$, we obtain 2^{n-1} clauses over literals $L_{31}, ..., L_{3n}$ and their complements.

- By repeatedly repeating step 4 with appropriate substitution of literals, we obtain 2^{n-1} clauses over literals $L_{N1}, ..., L_{Nn}$ and their complements.

- Finally, the 2^{n-1} clauses over literals $L_{N1}, ..., L_{Nn}$ and their complements are absorbed into the negation of Clauses(G_n) and the negation of Clauses(G_n) is proven to be valid.

The number of new clauses produced in this proof is $O(2^n \cdot 2^n \cdot N) = O(2^{3n})$. On the other hand, a tree resolution (resolution where the refutation is presented as a tree and the length is defined as the number of nodes in the tree) of Clauses(G_n) needs at least $2^{n(n-1)}$ steps [19]. Since a tree resolution and DPLL are p-equivalent, it implies that DPLL does not p-simulate $\mathrm{CE_G}$.

3.2 Mutilated Chessboard

Let $P_{i,j}, Q_{j,i}$ with $1 \leq i \leq 2n, 1 \leq j \leq 2n-1$ be proposition symbols. Let Γ_n be the union of the following sets of clauses:

$$\{P_{1,1} \vee Q_{1,1}\}$$
$$\{P_{1,j} \vee P_{1,j+1} \vee Q_{1,j+1} \mid 1 \leq j \leq 2n-2\}$$
$$\{P_{1,2n-1} \vee Q_{1,2n}\}$$
$$\{P_{i,1} \vee Q_{i-1,1} \vee Q_{i,1}\} \text{ for } i = 2, ..., 2n-1$$
$$\{P_{i,j} \vee Q_{i,j} \vee Q_{i,j+1} \vee P_{i+1,j} \mid 2 \leq j \leq 2n-2\} \text{ for } i = 2, ..., 2n-1$$
$$\{P_{i,2n-1} \vee Q_{i-1,2n} \vee Q_{i,2n}\} \text{ for } i = 2, ..., 2n-1$$
$$\{P_{2n,2} \vee Q_{2n-1,1}\}$$
$$\{P_{2n,j} \vee P_{2n,j+1} \vee Q_{n,j+1} \mid 2 \leq j \leq 2n-1\};$$
$$\{P_{2n,2n-1} \vee Q_{2n,2n}\}$$

Let Δ_n be the union of the following sets of clauses:

$$\{\neg P_{i,j} \vee \neg P_{i,j+1} \mid 1 \leq i \leq 2n, 1 \leq j \leq 2n-2\}$$
$$\{\neg Q_{i,j} \vee \neg Q_{i+1,j} \mid 1 \leq i \leq 2n-2, 1 \leq j \leq 2n\}$$
$$\{\neg P_{i,j} \vee \neg Q_{i,j}, \neg P_{i,j} \vee \neg Q_{i,j+1} \mid 1 \leq i \leq 2n-1, 1 \leq j \leq 2n-1\}$$
$$\{\neg Q_{i,j} \vee \neg P_{i+1,j-1}, \neg Q_{i,j} \vee \neg P_{i+1,j} \mid 1 \leq i \leq 2n-1, 2 \leq j \leq 2n\}$$

Δ_n represents that no pairs of adjacent propositions are allowed to be true simultaneously. Let Π be the set of clauses

$$\{P_{1,2n-1} \vee Q_{1,2n}, P_{2n,1} \vee Q_{2n-1,1}, \neg P_{1,2n-1} \vee \neg Q_{1,2n}, \neg P_{2n,1} \vee \neg Q_{2n-1,1}\}$$

The clauses of Π are supposed to represent statements on the top-right and the bottom-left corners of the chessboard. Finally, let Γ'_n be the following set of clauses:

$$\Gamma_n \cup \Delta_n \cup \{\neg P_{1,2n-1}, \neg Q_{1,2n}, \neg P_{2n,1}, \neg Q_{2n-1,1}\} \setminus \Pi.$$

The mutilated chessboard principle states that Γ'_n is unsatisfiable. Without of loss of generality, we consider Γ''_n which is defined to be the result of applying unit resolution on Γ'_n. The proof of the negation of Γ''_n may proceed as follows.

- To start with, we expand on P_{11}, Q_{11} and then use absorption and simplification, we obtain $\neg P_{11} \wedge Q_{11}$ and $P_{11} \wedge \neg Q_{11}$.
- Let $S_k = \{P_{i,j}, Q_{i,j} \mid i + j = k\}$.

  At phase $k \leq 2n-2$, we expand on all propositions in S_{k+1} and obtain $\left(\begin{array}{c} \text{2k-m+1} \\ \text{m} \end{array} \right)$ clauses with m being $\frac{k+1}{2}$, after absorption and simplification.
- Let $c(a,b) = \left(\begin{array}{c} \text{a-b+1} \\ \text{b} \end{array} \right)$.

  At phase $k \in \{2n-1, 2n\}$, we expand on $S_{k+1} \setminus \{P_{1,2n-1}, Q_{1,2n}, P_{2n,1}, Q_{2n-1,1}\}$. The numbers of clauses remained after absorption and simplification are respectively $c(4n-4, n)$ and $c(4n-4, n-2)$.
- At phase $k \in \{2n+1, ..., 4n-4\}$, we expand on S_{k+1} and obtain $c(8n-2k-2, 2n-m-(-1)^k \cdot 2)$ clauses after absorption and simplification.
- At phase $4n-3$, we expand on $\{Q_{2n-2,2n}, P_{2n-1,2n-1}, Q_{2n-1,2n-1}, P_{2n,2n-2}\}$. There will be no clause left after expansion and absorption. This completes the proof.

The number of phases is $4n-3$, the maximum number of clauses remained after absorption and simplification at each phase is less than 2^{4n-4}, the maximum number of clauses produced for each clause at any phase is less than $2 \cdot 2^{4n-4}$. Hence the total number of clauses produced in the proof is of the order $2^{O(n)}$. This length matches the lower-bound of the resolution refutations of the problem $2^{\Omega(n)}$.

3.3 Pigeonhole

Let P_{ij} with $1 \leq i \leq n+1, 1 \leq j \leq n$ be proposition symbols. Let Γ_n and Δ_n be respectively the following sets of formulas:

$$\{P_{i1} \vee \cdots \vee P_{in} \mid 1 \leq i \leq n+1\};$$
$$\{P_{ik} \wedge P_{jk} \mid 1 \leq i < j \leq n+1, 1 \leq k \leq n\}.$$

The pigeonhole principle states that the conjunction of the formulas of Γ_n implies the disjunction of that of Δ_n. Let $\mathcal{P}_n^{n+1}$ denote the union of Δ_n and the negation of Γ_n. The negation of $\mathcal{P}_n^{n+1}$ can be proved by CE_G as follows.

For simplicity, when counting the number of steps, we omit the steps where an expansion can immediately be followed by an absorption. These steps are to be added later.

- At the first phase, n clauses are produced (those which are immediately absorbed into $\mathcal{P}_n^{n+1}$ are not counted) by the axiom and expansion on $P_{11}, ..., P_{1n}$. After simplification, the set $\{P_{11}, ..., P_{1n}\}$ remains.

- At the second phase, n new clauses are produced by expansion on $P_{21}, ..., P_{n+1,1}, ..., P_{2n}, ..., P_{n+1,n}$. After simplification, the following set of clauses remains.
$$\{\neg P_{21} \wedge \cdots \wedge \neg P_{n+1,1}, ..., \neg P_{2n} \wedge \cdots \wedge \neg P_{n+1,n}\}$$

- Let φ_y^x be $\neg P_{xy} \wedge \cdots \wedge \neg P_{n+1,y}$.
  At phase $k > 2$, by expansion on $P_{k,i}, P_{k+1,i}, ..., P_{n+1,i}$ where i ranges from 1 to n depending on the clause to be expanded, $\binom{\text{n}}{\text{k-2}} \times (n + 2 - k)$ new clauses are produced and simplified to the following set of $\binom{\text{n}}{\text{k-1}}$ clauses:
$$\{\varphi_{i_1}^k \wedge \cdots \wedge \varphi_{i_{k-1}}^k \mid 1 \leq i_1 < \cdots < i_{k-1} \leq n\}$$

- At phase $(n + 1)$, every clause can be expanded into a sequence of clauses that can immediately be absorbed. This completes the proof. The total number of new clauses in these $n + 1$ phases is therefore:
$$n + \sum_{i=2}^{n} \binom{\text{n}}{\text{i-2}} \times (n + 2 - i) = n \cdot 2^{n-1}$$

The above number is comparable to the number of steps for proving the formula with monotone resolution, an incomplete variant of resolution for this kind of formulas [4]. The actual number of new clauses produced in the above proof is at most n times the above number, since at each phase, a length m clause is expanded to a clause of length at most $m + n$. The total proof length is therefore less than $n^2 \cdot 2^n$ which is slightly a reduction to the reported bound of steps $O(n^3 \cdot 2^n)$ in a corresponding resolution refutation in [4] (cf. the consequence of Lemma 1 of the paper).

3.4 Implication Graph

Let G be a directed acyclic graph with fan-in 2, n vertices and a single sink vertex. The formula associated with G, IMP(G) has a variable x_i for each node i in G and the following clauses:

- for each source node i in G, the clause (x_i);

- for the sink node s in G, the clause $(\neg x_s)$;

- for every triple of nodes i, j and k such that the edges (i, k), $(j, k) \in G$, the clause $(\neg x_i \vee \neg x_j \vee x_k)$.

These formulas were originally from [1] and were modified to IMP$^*(G)$ for the purpose of proving that linear resolution and negative resolution do not p-simulate unrestricted resolution in [3]. For IMP$^*(G)$, two variables x_i, y_i are associated to each vertex. Correspondingly, IMP$^*(G)$ has the following clauses:

- for each source node i in G, the clause $(x_i \vee y_i)$;

- for the sink node s in G, the clauses $(\neg x_s)$ and $(\neg y_s)$;
- for every triple of nodes i, j and k such that the edges (i,k), $(j,k) \in G$, the clauses $(\neg x_i \vee \neg x_j \vee x_k \vee y_k)$, $(\neg x_i \vee \neg y_j \vee x_k \vee y_k)$, $(\neg y_i \vee \neg x_j \vee x_k \vee y_k)$, and $(\neg y_i \vee \neg y_j \vee x_k \vee y_k)$.

A proof of the negation of $\mathrm{IMP}^*(G)$ of length linear to the size of $\mathrm{IMP}^*(G)$ is as follows. To begin with (considering the sink node), 4 clauses $\neg x_s \wedge \neg y_s$, $\neg x_s \wedge y_s$, $x_s \wedge \neg y_s$, and $x_s \wedge y_s$ are produced. The absorption rule is then applied to the last three clauses, since x_s, y_s are in the negation of $\mathrm{IMP}^*(G)$. After three applications of the rule, only the first clause (which is of the form $\neg x_k \wedge \neg y_k$ with $k = s$) remains.

- For each clause of the form $\neg x_k \wedge \neg y_k$: if k is a source vertex, the absorption rule can be applied, otherwise, let i, j be the two nodes leading to k, i.e. $(x_i \wedge x_j \wedge \neg x_k \wedge \neg y_k)$, $(x_i \wedge y_j \wedge \neg x_k \wedge \neg y_k)$, $(y_i \wedge x_j \wedge \neg x_k \wedge \neg y_k)$, and $(y_i \wedge y_j \wedge \neg x_k \wedge \neg y_k)$ are clauses in the negation of $\mathrm{IMP}^*(G)$.
- By expanding $\neg x_k \wedge \neg y_k$ on literals x_i, y_i, x_j and y_j, we obtain 16 clauses. Each of the clauses contains either $x_i \wedge x_j$, $x_i \wedge y_j$, $y_i \wedge x_j$, $y_i \wedge y_j$, $\neg x_i \wedge \neg y_i$, or $\neg x_j \wedge \neg y_j$ as a sub-clause.
- After applying the absorption rule to those clauses containing $x_i \wedge x_j$, $x_i \wedge y_j$, $y_i \wedge x_j$, or $y_i \wedge y_j$ and applying the simplification rule to those clauses containing $\neg x_i \wedge \neg y_i$ or $\neg x_j \wedge \neg y_j$, we obtain the two clauses $\neg x_i \wedge \neg y_i$ and $\neg x_j \wedge \neg y_j$.
- By moving carefully from the sink up to the source vertices level by level as demonstrated in the previous steps, all the produced clauses will eventually be absorbed into the negation of $\mathrm{IMP}^*(G)$ with the number of steps linear to the size of $\mathrm{IMP}^*(G)$.

4 Refutation System

A refutation system is a proof system for proving the unsatisfiability of a formula. A refutation system is sound if every rule of the system maps a (possibly empty) set of unsatisfiable formulas to an unsatisfiable formula. In order to make the system more convenient to use in certain cases and to compare $\mathrm{CE_G}$ with DPLL, we shall formulate the proof system $\mathrm{CE_G}$ as a refutation system for CNF formulas.

A formula is in CNF, if it is a conjunction of disjunctions of literals. A CNF-formula can also be represented by a set of clauses. Let Δ be a DNF-formula $\{c_1, ..., c_n\}$. The negation of Δ can be transformed to a CNF-formula by simply replacing each occurrence of the conjunction symbol in the clauses with the disjunction symbol and each literal with its complement. We use $\square$ to represent the empty clause which is never satisfiable. Let Γ, Π be sets of clauses. The rules are as follows:

Init: $$\Gamma \quad ; \square$$

Expansion: $$\frac{\Gamma \quad ; \quad \Pi, \varphi}{\Gamma \quad ; \quad \Pi, A \vee \varphi, \neg A \vee \varphi}$$

Simplification: $$\frac{\Gamma \quad ; \quad \Pi, \varphi \vee \psi}{\Gamma \quad ; \quad \Pi, \varphi}$$

Absorption: $$\frac{\Gamma, \varphi \quad ; \quad \Pi, \varphi}{\Gamma, \varphi \quad ; \quad \Pi}$$

Let us denote this system by $\mathrm{CE}_{\mathrm{NG}}$ (negated goal-oriented CE). A proof of the unsatisfiability of a CNF-formula Γ starts from the empty clause $\square$ and is finished when all clauses expanded from $\square$ are absorbed into Γ.

4.1 Complexity Considerations

For complexity considerations, we compare $\mathrm{CE}_{\mathrm{NG}}$ with DPLL based on the concept of p-simulation [6]. We first provide a few definitions.

Definition 4.1 *Let P be a refutation system. A P-refutation of a formula φ is a proof of the unsatisfiability of φ using the rules of P.*

Definition 4.2 *A refutation system P p-simulates Q, if there is a polynomial p such that for every P-refutation of a formula φ with refutation-size n, we can find a Q-refutation of φ with refutation-size $p(n)$.*

The refutation-size is the number of symbols in a refutation. Practically, we may use the number of steps as the measure, since normally the size and the number of steps are polynomially related in a refutation.

Definition 4.3 *Two refutation systems are p-equivalent, if they p-simulate each other.*

We shall show that $\mathrm{CE}_{\mathrm{NG}}$ p-simulates DPLL and on the other hand, DPLL does not p-simulate $\mathrm{CE}_{\mathrm{NG}}$. Let Γ be a set of clauses, and L be a literal. Let $\Gamma|L$ be the following set of formulas:

$$\{\varphi \mid \varphi \in \Gamma, L \notin \varphi, \neg L \notin \varphi\} \cup \{\varphi \mid \neg L \vee \varphi \in \Gamma\}$$

Let 1 denote satisfiability and 0 denote unsatisfiability. The procedure dpll(Γ) can be formulated as follows:

1. If Γ is empty, return 1;
2. If Γ contains the empty clause, return 0;
3. If Γ contains a pure literal L, return dpll($\Gamma|L$);
4. If Γ contains a unit clause L, return dpll($\Gamma|L$);
5. Otherwise, choose a literal L for branching;
6. If dpll($\Gamma|L$)=1 or dpll($\Gamma|\neg L$)=1, return 1; Otherwise, return 0.

From a DPLL-refutation of Γ, we can construct a refutation tree where each node is a point of branching (i.e. an application of step 5). We mark each node of the tree by the sequence of literals which have already been eliminated prior to the branching. A leaf node of a complete DPLL-refutation tree is marked by the literals needed for closing the branch. The literals in such a leaf node falsifies some clause of Γ. We can construct a $\mathrm{CE}_{\mathrm{NG}}$-refutation of Γ from a DPLL-refutation tree of Γ as follows.

Each branching corresponds to an application of the expansion rule. Without loss of generality, we assume that all applications of unit propagation (i.e. step 4) occur after a branching. A partial DPLL-refutation tree starting from the root with n leaf nodes (not necessarily representing a closed branch) can be represented by $\Gamma; \varphi_1, ..., \varphi_n$ where φ_i is the disjunction of the negation of literals of node i.

Consider a node marked with $L_1, ..., L_{m-1}$. The corresponding formula in the $\mathrm{CE}_{\mathrm{NG}}$ is $\neg L_1 \vee \cdots \vee \neg L_{m-1}$. For convenience, we write $\Delta(m)$ for $L_1, ..., L_{m-1}$ and $\varphi(m)$ for $\neg L_1 \vee \cdots \vee \neg L_{m-1}$. Let $\Gamma; \Pi, \varphi(m)$ be a representation of the partial DPLL refutation tree in which one leaf node is marked by $\Delta(m)$. Suppose that in the DPLL-refutation process, the two sub-nodes

$$\Delta(m), L_m, M_1, ..., M_{k_0} \text{ and } \Delta(m), \neg L_m, N_1, ..., N_{k_1}$$

are produced from the node $\Delta(m)$ by branching on the literal L_m. Producing these two nodes from $\Delta(m)$ corresponds to a proof of

$$\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1 \vee \cdots \vee \neg M_{k_0},\ \varphi(m) \vee L_m \vee \neg N_1 \vee \cdots \vee \neg N_{k_1}$$

from $\Gamma; \Pi, \varphi(m)$. This can be argued as follows:

1. From $\Gamma; \Pi, \varphi(m)$,

   we construct $\Gamma; \Pi, \varphi(m) \vee \neg L_m, \varphi(m) \vee L_m$.

2. If $k_0 = k_1 = 0$, we are done.

   If $k_0 = 0$ and $k_1 > 0$, go to step 5.

3. $k_0 > 0$. Since the node $\Delta(m), L_m, M_1, ..., M_{k_0}$ follows immediately after the node $\Delta(m)$, the literals $M_1, ..., M_{k_0}$ are unit-clauses produced by (repeated) unit propagation.

   Since M_1 is produced by unit propagation on L_m, some clause of Γ can be falsified by $\Delta(m), L_m, \neg M_1$. This means that $\varphi(m) \vee \neg L_m \vee M_1$ can be eliminated by the absorption rule.

   By an application of the expansion rule followed by an application of the absorption rule, we obtain

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1, \varphi(m) \vee L_m$ from

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m, \varphi(m) \vee L_m$.

4. Similarly, we can obtain

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1 \vee \cdots \vee \neg M_{i+1}, \varphi(m) \vee L_m$ from

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1 \vee \cdots \vee \neg M_i, \varphi(m) \vee L_m$ for $i = 1, ..., k_0 - 1$.

   This means that we can obtain

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1 \vee \cdots \vee \neg M_{k_0}, \varphi(m) \vee L_m$ from

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m, \varphi(m) \vee L_m$ by k_0 steps.

5. Finally, if $k_1 > 0$, the same argument can be used to obtain

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1 \vee \cdots \vee \neg M_{k_0}, \varphi(m) \vee L_m \vee \neg N_1 \vee \cdots \vee \neg N_{k_1}$ from

   $\Gamma; \Pi, \varphi(m) \vee \neg L_m \vee \neg M_1 \vee \cdots \vee \neg M_{k_0}, \varphi(m) \vee L_m$.

As demonstrated, each branching step and the following unit propagation in a DPLL-refutation can be transformed to a sequence of applications of $\mathrm{CE}_{\mathrm{NG}}$-rules, such that a complete DPLL-refutation tree with k leaf nodes corresponds to $\Gamma; \varphi_1, ..., \varphi_k$ where each φ_i can be absorbed into Γ. This proves the following theorem.

Theorem 4.1 *The proof system* $\mathrm{CE}_{\mathrm{NG}}$ *p-simulates* DPLL.

On the other hand, we already have that DPLL does not p-simulate CE_N from the previous section. This directly implies that DPLL does not p-simulate $\mathrm{CE}_{\mathrm{NG}}$ and hence $\mathrm{CE}_{\mathrm{NG}}$ is strictly stronger than DPLL.

4.2 Decomposition and Parallelism

In the presence of parallel computing power, it could in some circumstances be important to have the possibility for decomposing a refutation-goal and solving subgoals independently. For DPLL, it is relatively straightforward to develop strategies for parallel algorithms. Several tools based on the use of parallel computation for solving satisfiability problems based on DPLL have been developed [2, 20, 23]. For resolution, it is not obvious how to decompose a refutation goal or the refutation process based on the resolution rule. However, parallelism can be successful for restricted resolution. For instance, parallel execution of logic programs based on SLD-resolution [13] has been implemented and the advantage of parallelism has been reported [16, 12]. Our refutation systems are suitable for parallelism. A strategy for parallel algorithms can be formulated as follows:

$$\text{Decomposition:} \quad \frac{\Gamma\ ;\ \Pi_0, \Pi_1}{\Gamma\ ;\ \Pi_0, \qquad\qquad \Gamma\ ;\ \Pi_1}$$

This rule provides a basis for decomposing a refutation goal into independent subgoals which can be solved on clustered or networked workstations separately.

5 First Order Extension

First order formulas are propositional formulas extended with predicates and quantifiers. In addition to the symbols of propositional logic, the alphabet of a first order language contains the following symbols: a set of variables $v_1, v_2, ...$; the quantifiers $\forall, \exists$; for every $n \geq 1$ a (possibly empty) set of n-ary relation symbols; for every $n \geq 1$ a (possibly empty) set of n-ary function symbols; a (possibly empty) set of constants.

A formula is in prenex normal form if it is divided into a quantifier free formula and a prefix of a sequence of quantifiers as follows:

$$Q_1 x_1 Q_2 x_2 \cdots Q_k x_k F$$

where Q_i is either $\forall$ or $\exists$ and F is a quantifier free formula. A first order formula can be transformed to prenex normal form using the following equivalences:

$\forall x.\varphi(x) \supset \psi$	$\equiv$	$\exists x.(\varphi(x) \supset \psi)$
$\exists x.\varphi(x) \supset \psi$	$\equiv$	$\forall x.(\varphi(x) \supset \psi)$
$\varphi \supset \forall x.\psi(x)$	$\equiv$	$\forall x.(\varphi(x) \supset \psi(x))$
$\varphi \supset \exists x.\psi(x)$	$\equiv$	$\exists x.(\varphi(x) \supset \psi(x))$
$\neg(\forall x.\varphi(x))$	$\equiv$	$\exists x.\neg\varphi(x)$
$\neg(\exists x.\varphi(x))$	$\equiv$	$\forall x.\neg\varphi(x)$
$\forall x.\varphi(x) \wedge \psi$	$\equiv$	$\forall x.(\varphi(x) \wedge \psi)$
$\exists x.\varphi(x) \wedge \psi$	$\equiv$	$\exists x.(\varphi(x) \wedge \psi)$
$\varphi \wedge \forall x.\psi(x)$	$\equiv$	$\forall x.(\varphi(x) \wedge \psi(x))$
$\varphi \wedge \exists x.\psi(x)$	$\equiv$	$\exists x.(\varphi(x) \wedge \psi(x))$
$\forall x.\varphi(x) \vee \psi$	$\equiv$	$\forall x.(\varphi(x) \vee \psi)$
$\exists x.\varphi(x) \vee \psi$	$\equiv$	$\exists x.(\varphi(x) \vee \psi)$
$\varphi \vee \forall x.\psi(x)$	$\equiv$	$\forall x.(\varphi(x) \vee \psi(x))$
$\varphi \vee \exists x.\psi(x)$	$\equiv$	$\exists x.(\varphi(x) \vee \psi(x))$

Let φ be a formula in prenex normal form. Such a formula can be transformed into a formula with only existential quantifiers by the following rule:

$$\exists x_1 ... x_k \forall x_{k+1}.F(x_1, ..., x_k, x_{k+1}) \rightarrow \exists x_1 ... x_k.F(x_1, ..., x_k, f(x_1, ..., x_k))$$

where f is a new function symbol. This is called skolemization. A formula is valid if and only if its skolemized prenex normal form formula is. Let F be a skolemized prenex normal formula as follows:

$$\exists x_1 ... x_k . G(x_1, ..., x_k).$$

$G(x_1, ..., x_k)$ can then be transformed into a set of clauses as in the propositional case. In such clauses, the variables are all implicitly quantified by existential quantifiers. Proving a formula can then be reduced to proving the validity of such a set of clauses (i.e. the disjunction of the clauses).

5.1 Initial Extension

The basis of the first order extension is the propositional system CE. Let $x_1, ..., x_n$ be variables and $c_1, ..., c_n, d_1, ..., d_m$ ground terms. We may generalize the system straightforwardly as follows:

Axiom: $$A(c_1, ..., c_n), \neg A(c_1, ..., c_n)$$

Expansion: $$\frac{\Delta, \varphi(c_1, ..., c_n)}{\Delta, A(d_1, ..., d_m) \wedge \varphi(c_1, ..., c_n), \neg A(d_1, ..., d_m) \wedge \varphi(c_1, ..., c_n)}$$

Simplification: $$\frac{\Delta, \varphi(c_1, ..., c_n) \wedge \psi(d_1, ..., d_m)}{\Delta, \varphi(c_1, ..., c_n)}$$

Weakening: $$\frac{\Delta}{\Delta, \varphi(c_1, ..., c_n)}$$

Var-Introduction: $$\frac{\Delta, \varphi(c_1, ..., c_n)}{\Delta, \varphi(x_1, ..., x_n)} \quad (x_1, ..., x_n \notin \varphi(c_1, ..., c_n))$$

The restriction $x_1, ..., x_n \notin \varphi(c_1, ..., c_n)$ means that none of the variables $x_1, ..., x_n$ may occur in $\varphi(c_1, ..., c_n)$ This restriction in the variable introduction rule is necessary. Otherwise, we may infer $\varphi(x) \wedge \psi(x)$ from $\varphi(1) \wedge \psi(2)$. Let us denote this system by CE_0. A CE_0-proof can be rearranged to satisfy the following property: whenever a variable is introduced by the variable-introduction rule, it is absorbed into a clause at the next step.

Theorem 5.1 CE_0 *is sound and complete for first order DNF-formulas.*

The soundness is obvious, since the rules except the variable introduction are all the same as the rules of CE and the variable introduction replaces a formula with a corresponding existential formula which is weaker than the formula. For completeness, every valid DNF-formula can be proved with sufficiently many ground instances of each clause. Using the variable introduction rule, every instance of a clause can be absorbed into the clause. Therefore every valid DNF-formula can be proved using CE_0.

5.2 Further Development

Although the initial extension is complete, the efficiency of the system may not be desirable. For instance, a proof of the set of clauses

$$\{L(1), \neg L(x) \wedge L(e1x), \neg L(exexy) \wedge L(ee1xy), \neg L(e(e1)^n 11)\}$$

where $(e1)^0 1 = 1$ and $(e1)^k 1 = e1(e1)^{k-1}1$ requires at least 2^n steps, since the number of ground instances of $\neg L(x) \wedge L(e1x)$ needed for validating the set of clauses is 2^n (the reader is referred to [22] for details). To be able to prove such clauses more efficiently, we extend the system to be as follows. Let $t_1, ..., t_n$ be terms (may contain variables).

Axiom: $$A(x_1, ..., x_n), \neg A(x_1, ..., x_n)$$

Expansion: $$\frac{\Delta, \varphi(x_1, ..., x_n)}{\Delta, A(y_1, ..., y_m) \wedge \varphi(x_1, ..., x_n), \neg A(y_1, ..., y_m) \wedge \varphi(x_1, ..., x_n)}$$

Simplification: $$\frac{\Delta, \varphi(x_1, ..., x_n) \wedge \psi(y_1, ..., y_m)}{\Delta, \varphi(x_1, ..., x_n)}$$

Weakening: $$\frac{\Delta}{\Delta, \varphi(x_1, ..., x_n)}$$

Var-Introduction: $$\frac{\Delta, \varphi(t_1, ..., t_n)}{\Delta, \varphi(x_1, ..., x_n)} \quad (x_1, ..., x_n \notin \varphi(t_1, ..., t_n))$$

Note that $A(x_1, ..., x_n)$ is an atomic formula, but A is not necessarily a predicate symbol, for instance, $A(x_1, ..., x_n)$ may represent the atomic formula $P(f(x_1))$. The variables $y_1, ..., y_m$ and $x_1, ..., x_n$ in the expansion rule may overlap. The variables $y_1, ..., y_m$ and $x_1, ..., x_n$ in the simplification rule may overlap as well. Since variables may be introduced in the axiom and expansion rule, we may ask whether it is necessary to keep the variable introduction rule in this system. Unfortunately, the system is not complete without the variable introduction rule. For instance, the validity of set of clauses $\{A(a), \neg A(x)\}$ is not provable without the variable introduction rule. Let us denote the system by CE_1.

Theorem 5.2 CE_1 *is sound and complete for first order DNF-formulas.*

The soundness of the rules is obvious. The completeness follows from the completeness of CE_0. The benefit of CE_1 over CE_0 is the improved efficiency. This is to be explained in Section 5.4.

5.3 Goal Oriented Proofs

Similar to the propositional case, we can explicitly incorporate the proof goal into the system. Let us denote the axiom by I, the expansion rule by E, the simplification rule by S, the absorption rule by A, and the variable introduction rule by V. The rules are as follows:

I: $\quad \Lambda \;;\; A(x_1,...,x_n), \neg A(x_1,...,x_n)$

E: $$\frac{\Lambda \;;\; \Delta, \varphi(x_1,...,x_n)}{\Lambda \;;\; \Delta, A(y_1,...,y_m) \wedge \varphi(x_1,...,x_n), \neg A(y_1,...,y_m) \wedge \varphi(x_1,...,x_n)}$$

S: $$\frac{\Lambda \;;\; \Delta, \varphi(x_1,...,x_n) \wedge \psi(y_1,...,y_m)}{\Lambda \;;\; \Delta, \varphi(x_1,...,x_n)}$$

A: $$\frac{\Lambda, \varphi \;;\; \Delta, \varphi}{\Lambda, \varphi \;;\; \Delta}$$

V: $$\frac{\Lambda \;;\; \Delta, \varphi(t_1,...,t_n)}{\Lambda \;;\; \Delta, \varphi(x_1,...,x_n)} \quad (x_1,...,x_n \notin \varphi(t_1,...,t_n))$$

This formulation makes the goal explicit, so that we can compare the left hand side and the right hand side of the semicolon in order to decide what to do in a given step. For instance, whenever there is a clause $\varphi(x_1,...,x_n)$ in the left hand side of the semicolon, we can delete any clause in the right hand side of the semicolon that partially matches $\varphi(x_1,...,x_n)$ by applying the simplification rule, the variable introduction rule and the absorption rule. This can be formulated as a derived absorption rule as follows:

B: $$\frac{\Lambda, \varphi(x_1,...,x_n) \;;\; \Delta, \varphi(t_1,...,t_n) \wedge \psi(s_1,...,s_m)}{\Lambda, \varphi(x_1,...,x_n) \;;\; \Delta}$$

5.4 Complexity Considerations

For complexity considerations, we compare CE_1 and CE_0. Since the former is an extension of the latter, CE_1 p-simulates CE_0. On the other hand, we shall show that CE_0 does not p-simulate CE_1 in the framework of goal oriented proofs. Let Λ_n be the set of formulas:

$$\{L(1), \neg L(x) \wedge L(e1x), \neg L(exexy) \wedge L(ee1xy), \neg L(e(e1)^n 11)\}.$$

As mentioned earlier, a proof of Λ_n by CE_0 requires at least 2^n steps. On the other hand, a proof of Λ_n by the goal oriented version of CE_1 can be carried out within $O(n)$ steps as follows:

- To start with, by the axiom, we have the set of clauses $\{L(1), \neg L(1)\}$. The first one can be absorbed into Λ_n.
- We then expand the remaining clause on $L(e(e1)^n 11)$ and obtain $\{\neg L(1) \wedge L(e(e1)^n 11), \neg L(1) \wedge \neg L(e(e1)^n 11)\}$. The second one can be absorbed into Λ_n by applying the derived absorption rule.
- By applying the variable introduction rule to the remaining clause, we obtain $\neg L(y) \wedge L(e(e1)^n 1y)$.
- We expand the clause $\neg L(y) \wedge L(e(e1)^n 1y)$ on $L(e(e1)^{n-1}1e(e1)^{n-1}1y)$ and obtain $\{$ $\neg L(y) \wedge L(e(e1)^n 1y) \wedge L(e(e1)^{n-1}1e(e1)^{n-1}1y)$, $\neg L(y) \wedge L(e(e1)^n 1y) \wedge \neg L(e(e1)^{n-1}1e(e1)^{n-1}1y)$ $\}$. The second one can be absorbed into Λ_n by the derived absorption rule.
- We then expand the remaining clause on $L(e(e1)^{n-1}1y)$. After simplification and variable introduction, we obtain $\neg L(y) \wedge L(e(e1)^{n-1}1y)$.

- By an inductive argument, we can reach $\neg L(y) \wedge L(e1y)$ within $O(n)$ steps. This formulas is then absorbed into Λ_n.

The following theorem follows from the above discussion.

Theorem 5.3 *The proof system* CE_0 *does not p-simulate* CE_1.

5.5 Discussion

There are four rules that may introduce new variables in CE_1, namely, the axiom, the expansion rule, the weakening rule and the variable introduction rule. We have explained that it is necessary with the variable introduction rule, otherwise, the system is not complete. On the other hand, the system is still complete if we restrict the axiom, the expansion rule and the variable introduction rule in such a way that no variables are allowed in the new literals. This would make the system much simpler. The question is whether it is desirable to make such a restriction.

Definition 5.1 *A variable-restricted proof is a proof in* CE_1 *in which every occurrence of any variable is introduced by the use of the variable introduction rule.*

Let CE_* denote the system for variable-restricted proofs. CE_0-proofs are special cases of variable-restricted proofs. The axiom in CE_1 can be considered as a derived rule in CE_*, since the axiom in CE_1 corresponds a combination of a propositional axiom and an application of variable introduction. The weakening rule in CE_1 can also be considered as a derived rule in CE_*. The expansion rule in CE_1 is however not a derived rule in CE_*. This is explained in detail in the following paragraphs. We first introduce the concept of regular proofs. Since the destination of a clause is either expansion or absorption, a proof may follow a pattern such as:

- Before an expansion, we apply the simplification rule and variable introduction (several times, since several clauses may be simplified and generalized to one clause);
- Before an absorption, we also apply the simplification rule and variable introduction (represented by the derived absorption rule).

Let $S(\varphi)$ denote an application of simplification with (an instance) of φ being the new clause, $V(\varphi)$ denote an application of variable introduction with of φ being the new clause, and $E(\varphi)$ denote an application of the expansion rule with φ as the clause being expanded. Let $S^*(\varphi)$ denote a (possibly empty) sequence of applications of $S(\varphi)$ and $V^*(\varphi)$ denote a (possibly empty) sequence of applications of $V(\varphi)$. If a set of clauses is valid, we can construct a proof of the set of clauses with the following sequence of applications of the proof rules:

$$A,\ S^*(\varphi_1),\ V^*(\varphi_1),\ E(\varphi_1),\ ...,\ S^*(\varphi_n),\ V^*(\varphi_n),\ E(\varphi_n),\ B', ..., B'.$$

Let us call such a proof a regular proof and the length of a regular proof be the length of such a proof where every application of the derived rules is replaced by the corresponding combinations of the original rules. Restructuring CE_*-proof to a regular proof does not increase the proof length and we obtain the following lemma.

Lemma 5.1 *If there is a* CE_**-proof of* Λ *with length* n*, there is a regular proof of* Λ *with length* $\leq n$.

Lemma 5.2 *If there is a* CE_**-proof of* Λ *with length* n*, we can construct a* CE_0*-proof of* Λ *with length* $\leq 2n$*.*

Proof: We first reorganize a CE_*-proof into a regular proof A, $S^*(\varphi_1)$, $V^*(\varphi_1)$, $E(\varphi_1)$, ..., $S^*(\varphi_m)$, $V^*(\varphi_m)$, $E(\varphi_m)$, $B', ..., B'$. Between A and $E(\varphi_1)$, we delete the applications of the variable introduction rule (if any), and reintroduce two applications of the variable introduction rule after $E(\varphi_1)$ if it is necessary. The length of the whole proof may be increased by 1, since we may need the variable introduction rule twice after $E(\varphi_1)$ when sometimes it is sufficient with one application of the rule before $E(\varphi_1)$. This new proof is not necessarily a regular proof. If it is not, we reorganize the new proof into a regular proof and apply the same strategy to φ_2. This procedure is repeated until all applications of the variable introduction rule are placed after the expansion on φ_m. The length of the new proof is at most $n + m < 2n$. QED.

Theorem 5.4 CE_* *does not p-simulates* CE_1*.*

Corollary 5.1 *The expansion rule of* CE_1 *is not a derived rule of* CE_**.*

6 Concluding Remarks

Propositional clause expansion proof systems CE and $\mathrm{CE_G}$ for DNF-formulas have been presented. Based on these systems, the refutation systems $\mathrm{CE_{NG}}$ for proving the unsatisfiability of CNF-formulas was discussed. Finally, first order extensions of CE, denoted by CE_0 and CE_1, were given.

The application examples have shown that the system is easy to use for proving hard instances of valid formulas. The propositional clause expansion proof systems are stronger than DPLL, in the sense that DPLL does not p-simulate $\mathrm{CE_G}$. On the other hand, comparing with resolution, it could be easier to make guidelines for assessing the usefulness of the new clauses while keeping the capability of constructing short proofs, since there are many clauses that can be used as intermediate goals for an application of the rules of the systems. The approach also has better potential for utilizing parallel computing power. A few simple guidelines for the system $\mathrm{CE_G}$ were given. Such guidelines deserve further investigation.

The extension CE_1 is not a simple lifting of the proof system CE_0 with derived rules for generalized variable introduction. A comparison of CE_1 and CE_0 shows that the extension is capable of producing significantly shorter proofs than proofs solely based on using ground instances of first order clauses.

References

[1] E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. In *31st Annual ACM Symposium on Theory of Computing*, pages 517–526. ACM, 1999.

[2] M. Bøhm and E. Speckenmeyer. A fast parallel sat-solver - efficient workload balancing. *Annals of Mathematics and Artificial Intelligence*, 17(3-4):381–400, 1996.

[3] J. Buresh-Oppenheim, D. Mitchell, and T. Pitassi. Linear and negative resolution are weaker than resolution. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(074), 2001.

[4] S. R. Buss and T. Pitassi. Resolution and the weak pigeonhole principle. In *Lecture Notes in Computer Science 1414 (CSL 97)*, pages 149–156. Springer-Verlag, 1998.

[5] C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

[6] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44:36–50, 1979.

[7] S. S. Dantchev and S. Riis. Planar tautologies hard for resolution. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 220–229. IEEE, 2002.

[8] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of ACM*, 5:394–397, 1962.

[9] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of ACM*, 7(3):201–215, 1960.

[10] E. Giunchiglia, M. Maratea, and A. Tacchella. Dependent and independent variables in propositional satisfiability. In *Lecture Notes in Computer Science 2424 (JELIA 02)*, pages 296–307. Springer-Verlag, 2002.

[11] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[12] P. Kefalas and I. P. Vlahava. Multiple or-parallel resolution: Meta-level control of parallel logic programs. In *Lecture Notes in Computer Science 1123 (Euro-Par 96)*, pages 694–703. Springer-Verlag, 1996.

[13] R. A. Kowalski. Predicate logic as programming language. In *Proceedings of IFIP Congress 74*, pages 569–574. North-Holland, 1996.

[14] C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *15th Int'l Joint Conf. on Artificial Intelligence*, pages 366–371, 1997.

[15] P. Liberatore. On the complexity of choosing the branching literal in dpll. *Artificial Intelligence*, 116:315–326, 2000.

[16] E. L. Lusk, S. Mudambi, R. A. Overbeek, and P. Szeredi. Applications of the aurora parallel prolog system to computational molecular biology. In *Proc. 1993 International Symposium on Logic Programming*, pages 353–369. MIT Press, 1993.

[17] J. A. Robinson. A machine oriented logic based on the resolution principle. *Journal of ACM*, 12:23–41, 1965.

[18] G. S. Tseitin. On the complexity of derivation in propositional logic. In *Automated Reasoning*, pages 466–483. New York: Springer-Verlag, 1983.

[19] A. Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[20] H. Zhang, M. P. Bonacina, and J. Hsiang. Psato: a distributed propositional prover and its application to quasigroup problems. *Journal of Symbolic Computation*, 21(4):543–560, 1996.

[21] H. Zhang and M. E. Stickel. Implementing the davis-putnam method. *Journal of Automated Reasoning*, 24:277–296, 2000.

[22] W. Zhang. Cut elimination and automatic proof procedures. *Theoretical Computer Science*, 91(2):265–284, 1991.

[23] W. Zhang, Z. Huang, and J. Zhang. Parallel execution of stochastic search procedures on reduced sat instances. In *Lecture Notes in Computer Science 2417 (PRICAI 02)*, pages 108–117. Springer-Verlag, 2002.